

Foundation of Cryptography (0368-4162-01), Lecture 7

MACs and Signatures

Iftach Haitner, Tel Aviv University

December 27, 2011

Section 1

Message Authentication Code (MAC)

Message Authentication Code (MAC)

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

- 1 Gen(1^n) outputs a key $k \in \{0, 1\}^*$
- 2 Mac(k, m) outputs a "tag" t
- 3 Vrfy(k, m, t) output 1 (YES) or 0 (NO)

Message Authentication Code (MAC)

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

- 1 Gen(1^n) outputs a key $k \in \{0, 1\}^*$
- 2 Mac(k, m) outputs a "tag" t
- 3 Vrfy(k, m, t) output 1 (YES) or 0 (NO)

Consistency: Vrfy $_k(m, t) = 1$ for any $k \in \text{Supp}(\text{Gen}(1^n))$,
 $m \in \{0, 1\}^n$ and $t = \text{Mac}_k(m)$

Message Authentication Code (MAC)

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

- ① Gen(1^n) outputs a key $k \in \{0, 1\}^*$
- ② Mac(k, m) outputs a "tag" t
- ③ Vrfy(k, m, t) output 1 (YES) or 0 (NO)

Consistency: $\text{Vrfy}_k(m, t) = 1$ for any $k \in \text{Supp}(\text{Gen}(1^n))$, $m \in \{0, 1\}^n$ and $t = \text{Mac}_k(m)$

Definition 2 (Existential unforgeability)

A MAC (Gen, Mac, Vrfy) is existential unforgeable (EU), if for any oracle-aided PPT A:

$$\Pr[k \leftarrow \text{Gen}(1^n); (m, t) \leftarrow A^{\text{Mac}_k, \text{Vrfy}_k}(1^n): \\ \text{Vrfy}_k(m, t) = 1 \wedge \text{Mac}_k \text{ was not asked on } m] = \text{neg}(n)$$

- “Private key” definition

- “Private key” definition
- Security definition too strong?

- “Private key” definition
- Security definition too strong? Any message? Use of Verifier?

- “Private key” definition
- Security definition too strong? Any message? Use of Verifier?
- “Replay attacks”

- “Private key” definition
- Security definition too strong? Any message? Use of Verifier?
- “Replay attacks”

Length-restricted MACs

Definition 3 (Length-restricted MAC)

Same as in Definition 1, but for $k \in \text{Supp}(G(1^n))$, Mac_k and Vrfy_k only accept messages of length n .

Bounded-query MACs

Definition 4 (ℓ -time MAC)

A MAC scheme is existential unforgeable against ℓ queries (for short, ℓ -time MAC), if it is existential unforgeable as in Definition 2, but A can only ask for ℓ queries.

Section 2

Constructions

Zero-time, restricted length, MAC

Construction 5 (Zero-time, restricted length, MAC)

- $\text{Gen}(1^n)$: outputs $k \leftarrow \{0, 1\}^n$
- $\text{Mac}_k(m) = k$
- $\text{Vrfy}_k(m, t) = 1$, iff $t = k$

Zero-time, restricted length, MAC

Construction 5 (Zero-time, restricted length, MAC)

- $\text{Gen}(1^n)$: outputs $k \leftarrow \{0, 1\}^n$
- $\text{Mac}_k(m) = k$
- $\text{Vrfy}_k(m, t) = 1$, iff $t = k$

Claim 6

The above scheme is a length-restricted, zero-time MAC

ℓ -wise independent hash

Definition 7 (ℓ -wise independent)

A function family \mathcal{H} from $\{0, 1\}^n$ to $\{0, 1\}^m$ is ℓ -wise independent, where $\ell \in \mathbb{N}$, if for every distinct $x_1, \dots, x_\ell \in \{0, 1\}^n$ and every $y_1, \dots, y_\ell \in \{0, 1\}^m$, it holds that $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge \dots \wedge h(x_\ell) = y_\ell] = 2^{-\ell m}$.

ℓ -times, restricted length, MAC

Construction 8 (ℓ -time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ be an efficient $(\ell + 1)$ -wise independent function family.

- $\text{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\text{Mac}(h, m) = h(m)$
- $\text{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

ℓ -times, restricted length, MAC

Construction 8 (ℓ -time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ be an efficient $(\ell + 1)$ -wise independent function family.

- $\text{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\text{Mac}(h, m) = h(m)$
- $\text{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

Claim 9

The above scheme is a length-restricted, ℓ -time MAC

ℓ -times, restricted length, MAC

Construction 8 (ℓ -time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ be an efficient $(\ell + 1)$ -wise independent function family.

- $\text{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\text{Mac}(h, m) = h(m)$
- $\text{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

Claim 9

The above scheme is a length-restricted, ℓ -time MAC

Proof: HW

OWF \implies existential unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ instead of \mathcal{H} .

Claim 11

Assuming that \mathcal{F} is a PRF, then Construction 10 is an existential unforgeable MAC.

OWF \implies existential unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ instead of \mathcal{H} .

Claim 11

Assuming that \mathcal{F} is a PRF, then Construction 10 is an existential unforgeable MAC.

Proof:

OWF \implies existential unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n: \{0, 1\}^n \mapsto \{0, 1\}^n\}$ instead of \mathcal{H} .

Claim 11

Assuming that \mathcal{F} is a PRF, then Construction 10 is an existential unforgeable MAC.

Proof: Easy to prove if \mathcal{F} is a family of random functions.
Hence, also holds in case \mathcal{F} is a PRF. \square

Any Length

Collision Resistant Hash Family

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^* \mapsto \{0, 1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow A(1^n, h): x \neq x' \in \{0, 1\}^* \\ \wedge h(x) = h(x')] = \text{neg}(n)$$

for any PPT A .

Any Length

Collision Resistant Hash Family

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^* \mapsto \{0, 1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow A(1^n, h): x \neq x' \in \{0, 1\}^* \\ \wedge h(x) = h(x')] = \text{neg}(n)$$

for any PPT A .

- Not known to be implied by OWF

Any Length

Length restricted MAC \implies MAC**Construction 13 (Length restricted MAC \implies MAC)**

Let $(\text{Gen}, \text{Mac}, \text{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^* \mapsto \{0, 1\}^n\}$ be an efficient function family.

- $\text{Gen}'(1^n): k \leftarrow \text{Gen}(1^n), h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\text{Mac}'_{k,h}(m) = \text{Mac}_k(h(m))$
- $\text{Vrfy}'_{k,h}(t, m) = \text{Vrfy}_k(t, h(m))$

Any Length

Length restricted MAC \implies MAC**Construction 13 (Length restricted MAC \implies MAC)**

Let $(\text{Gen}, \text{Mac}, \text{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^* \mapsto \{0, 1\}^n\}$ be an efficient function family.

- $\text{Gen}'(1^n): k \leftarrow \text{Gen}(1^n), h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\text{Mac}'_{k,h}(m) = \text{Mac}_k(h(m))$
- $\text{Vrfy}'_{k,h}(t, m) = \text{Vrfy}_k(t, h(m))$

Claim 14

Assume \mathcal{H} is an efficient collision-resistant family and $(\text{Gen}, \text{Mac}, \text{Vrfy})$ is existential unforgeable, then $(\text{Gen}', \text{Mac}', \text{Vrfy}')$ is existential unforgeable MAC.

Any Length

Length restricted MAC \implies MAC**Construction 13 (Length restricted MAC \implies MAC)**

Let $(\text{Gen}, \text{Mac}, \text{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n: \{0, 1\}^* \mapsto \{0, 1\}^n\}$ be an efficient function family.

- $\text{Gen}'(1^n): k \leftarrow \text{Gen}(1^n), h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\text{Mac}'_{k,h}(m) = \text{Mac}_k(h(m))$
- $\text{Vrfy}'_{k,h}(t, m) = \text{Vrfy}_k(t, h(m))$

Claim 14

Assume \mathcal{H} is an efficient collision-resistant family and $(\text{Gen}, \text{Mac}, \text{Vrfy})$ is existential unforgeable, then $(\text{Gen}', \text{Mac}', \text{Vrfy}')$ is existential unforgeable MAC.

Proof: ?

Section 3

Signature Schemes

Definition

Definition 15 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

- 1 Gen(1^n) outputs a pair of keys $(s, v) \in \{0, 1\}^* \times \{0, 1\}^*$
- 2 Sign(s, m) outputs a "signature" $\sigma \in \{0, 1\}^*$
- 3 Vrfy(v, m, σ) outputs 1 (YES) or 0 (NO)

Definition

Definition 15 (Signature schemes)

A trippet of PPT's $(\text{Gen}, \text{Sign}, \text{Vrfy})$ such that

- 1 $\text{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0, 1\}^* \times \{0, 1\}^*$
- 2 $\text{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0, 1\}^*$
- 3 $\text{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

Consistency: $\text{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \text{Supp}(\text{Gen}(1^n))$, $m \in \{0, 1\}^*$ and $\sigma \in \text{Supp}(\text{Sign}_s(m))$

Definition

Definition 15 (Signature schemes)

A trippet of PPT's $(\text{Gen}, \text{Sign}, \text{Vrfy})$ such that

- ① $\text{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0, 1\}^* \times \{0, 1\}^*$
- ② $\text{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0, 1\}^*$
- ③ $\text{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

Consistency: $\text{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \text{Supp}(\text{Gen}(1^n))$, $m \in \{0, 1\}^*$ and $\sigma \in \text{Supp}(\text{Sign}_s(m))$

Definition 16 (Existential unforgeability)

A signature scheme is existential unforgeable (EU), if for any oracle-aided PPT A

$$\Pr[(s, v) \leftarrow \text{Gen}(1^n); (m, \sigma) \leftarrow A^{\text{Sign}_s}(1^n, v): \\ \text{Vrfy}_v(m, \sigma) = 1 \wedge \text{Sign}_s \text{ was not asked on } m] = \text{neg}(n)$$

- Signature \implies MAC

- Signature \implies MAC
- “Harder” to construct than MACs: (even restricted forms) require OWF

- Signature \implies MAC
- “Harder” to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given

- Signature \implies MAC
- “Harder” to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong existential unforgeable signatures (for short, strong signatures): infeasible to generate *any* new valid signatures (even for message for which a signature was asked)

- Signature \implies MAC
- “Harder” to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong existential unforgeable signatures (for short, strong signatures): infeasible to generate *any* new valid signatures (even for message for which a signature was asked)

Theorem 17

OWFs imply strong existential unforgeable signatures.

Section 4

OWFs \Rightarrow Signatures

Length-restricted Signatures

Definition 18 (Length-restricted Signatures)

Same as in Definition 15, but for $(s, v) \in \text{Supp}(G(1^n))$, Sign_s and Vrfy_v only accept messages of length n .

Bounded-query Signatures

Definition 19 (ℓ -time signatures)

A signature scheme is existential unforgeable against ℓ -query (for short, ℓ -time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Bounded-query Signatures

Definition 19 (ℓ -time signatures)

A signature scheme is existential unforgeable against ℓ -query (for short, ℓ -time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Claim 20

Assuming CRH exists: length restricted, one-time signatures, imply one-time signatures.

OWF \Rightarrow length restricted, One Time Signature**Construction 21 (length restricted, one time signature)**

Let $f: \{0, 1\}^n \mapsto \{0, 1\}^n$.

- 1 $\text{Gen}(1^n)$: $s_1^0, s_1^1, \dots, s_n^0, s_n^1 \leftarrow \{0, 1\}^n$, let
 $s = (s_1^0, s_1^1, \dots, s_n^0, s_n^1)$ and
 $v = (v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \dots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1))$
- 2 $\text{Sign}(s, m)$: Output $(s_1^{m_1}, \dots, s_n^{m_n})$
- 3 $\text{Vrfy}(v, m, \sigma = (\sigma_1, \dots, \sigma_n))$ check that $f(\sigma_i) = v_{m_i}$ for all
 $i \in [n]$

OWF \implies length restricted, One Time Signature**Construction 21 (length restricted, one time signature)**

Let $f: \{0, 1\}^n \mapsto \{0, 1\}^n$.

- 1 **Gen**(1^n): $s_1^0, s_1^1, \dots, s_n^0, s_n^1 \leftarrow \{0, 1\}^n$, let
 $s = (s_1^0, s_1^1, \dots, s_n^0, s_n^1)$ and
 $v = (v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \dots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1))$
- 2 **Sign**(s, m): Output $(s_1^{m_1}, \dots, s_n^{m_n})$
- 3 **Vrfy**($v, m, \sigma = (\sigma_1, \dots, \sigma_n)$) check that $f(\sigma_i) = v_{m_i}$ for all
 $i \in [n]$

Lemma 22

Assume that f is a OWF, then scheme from Construction 21 is a length restricted one-time signature scheme

Proving Lemma 22

Let a PPT A , $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that break the security of Construction 21, we use A to invert f .

Algorithm 23 (Inv)

Input: $y \in \{0, 1\}^n$

- 1 Choose $(s, v) \leftarrow \text{Gen}(1^n)$ and replace $v_{j^*}^{j^*}$ for a random $i^* \in [n]$ and $j^* \in \{0, 1\}$, with y .
- 2 If $A(1^n, v)$ asks to sign message $m \in \{0, 1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
- 3 Let (m, σ) be A 's output. If σ is not a valid signature for m , or $m_{i^*} \neq j^*$, abort.
Otherwise, return σ_{j^*} .

Proving Lemma 22

Let a PPT A , $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that break the security of Construction 21, we use A to invert f .

Algorithm 23 (Inv)

Input: $y \in \{0, 1\}^n$

- 1 Choose $(s, v) \leftarrow \text{Gen}(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0, 1\}$, with y .
- 2 If $A(1^n, v)$ asks to sign message $m \in \{0, 1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
- 3 Let (m, σ) be A 's output. If σ is not a valid signature for m , or $m_{i^*} \neq j^*$, abort.
Otherwise, return σ_{j^*} .

v is distributed as it is in the real "signature game" (ind. of i^* and j^*).

Proving Lemma 22

Let a PPT A , $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that break the security of Construction 21, we use A to invert f .

Algorithm 23 (Inv)

Input: $y \in \{0, 1\}^n$

- ➊ Choose $(s, v) \leftarrow \text{Gen}(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0, 1\}$, with y .
- ➋ If $A(1^n, v)$ asks to sign message $m \in \{0, 1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
- ➌ Let (m, σ) be A 's output. If σ is not a valid signature for m , or $m_{i^*} \neq j^*$, abort.
Otherwise, return σ_{j^*} .

v is distributed as it is in the real "signature game" (ind. of i^* and j^*). Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 24 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 24 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)

Stateful schemes (also known as, Memory-dependant schemes)

Definition 24 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)
- We'll use it a building block for building a stateless scheme

Naive construction

Let $(\text{Gen}, \text{Sign}, \text{Vrfy})$ be a one-time signature scheme.

Construction 25 (Naive construction)

- 1 $\text{Gen}'(1^n)$ outputs $(s_1, v_1) = \text{Gen}(1^n)$.
- 2 $\text{Sign}_s(m_i)$, where m_i is i 'th message to sign:
 Let $((m_1, \sigma'_1), \dots, (m_{i-1}, \sigma'_{i-1}))$ be the previously signed pairs of messages/signatures.
 - 1 Let $(s_{i+1}, v_{i+1}) \leftarrow \text{Gen}(1^n)$
 - 2 Let $\sigma_i = \text{Sign}_{s_{i-1}}(m_i, v_{i+1})$, and output $\sigma'_i = (\sigma'_{i-1}, m_i, v_{i+1}, \sigma_i)$.
- 3 $\text{Vrfy}'_v(m, \sigma' = (m_1, v_2, \sigma_1), \dots, (m_i, v_{i+1}, \sigma_i))$:
 - 1 Verify $\text{Vrfy}_{v_{j-1}}((m_j, v_{j+1}), \sigma_j) = 1$ for every $j \in [i]$
 - 2 Verify $m_i = m$

- 1 State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
- 2 Inefficient scheme, though still polynomial, both running time and signature size are linear in number of signatures

- 1 State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
- 2 Inefficient scheme, though still polynomial, both running time and signature size are linear in number of signatures
- 3 Critically uses the fact that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ works for any length

Lemma 26

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Lemma 26

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Proof: Let a PPT A' , $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that breaks the security of $(\text{Gen}', \text{Sign}', \text{Vrfy}')$, we present a PPT A that breaks the security of $(\text{Gen}, \text{Sign}, \text{Vrfy})$.

Lemma 26

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Proof: Let a PPT A' , $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \text{poly}$ that breaks the security of $(\text{Gen}', \text{Sign}', \text{Vrfy}')$, we present a PPT A that breaks the security of $(\text{Gen}, \text{Sign}, \text{Vrfy})$.

- We assume for simplicity that p also bounds the query complexity of A'

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Claim 27

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma) \in [q]$ such that:

- 1 Sign' was *not* asked by A' on $m_{\tilde{i}}$.
- 2 Sign' was asked by A' on m_i , for every $i \in [\tilde{i} - 1]$

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Claim 27

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma) \in [q]$ such that:

- 1 Sign' was *not* asked by A' on $m_{\tilde{i}}$.
- 2 Sign' was asked by A' on m_i , for every $i \in [\tilde{i} - 1]$

Proof:

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Claim 27

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma) \in [q]$ such that:

- 1 Sign' was *not* asked by A' on $m_{\tilde{i}}$.
- 2 Sign' was asked by A' on m_i , for every $i \in [\tilde{i} - 1]$

Proof: Let \tilde{i} be the maximal index such that condition (2) holds (cannot be $q + 1$). \square

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Claim 27

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma) \in [q]$ such that:

- ① Sign' was *not* asked by A' on $m_{\tilde{i}}$.
- ② Sign' was asked by A' on m_i , for every $i \in [\tilde{i} - 1]$

Proof: Let \tilde{i} be the maximal index such that condition (2) holds (cannot be $q + 1$). \square

- Let $\tilde{m} = (m_{\tilde{i}}, v_{\tilde{i}+1})$, and let $s_{\tilde{i}}$ be the signing key generated together with $v_{\tilde{i}}$.

Proving Lemma 26 cont.

Let the random variables

$(m, \sigma = (m_1, v_2, \sigma_1), \dots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A'

Claim 27

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma) \in [q]$ such that:

- ① Sign' was *not* asked by A' on $m_{\tilde{i}}$.
- ② Sign' was asked by A' on m_i , for every $i \in [\tilde{i} - 1]$

Proof: Let \tilde{i} be the maximal index such that condition (2) holds (cannot be $q + 1$). \square

- Let $\tilde{m} = (m_{\tilde{i}}, v_{\tilde{i}+1})$, and let $s_{\tilde{i}}$ be the signing key generated together with $v_{\tilde{i}}$.
- Hence, $\text{Sign}_{s_{\tilde{i}}}(\sigma_{\tilde{i}}, \tilde{m}) = 1$, and $\text{Sign}_{s_{\tilde{i}}}$ was not queried by Sign'_s on \tilde{m} .

Definition of A

Algorithm 28 (A)

Input: $v, 1^n$

Oracle: Sign_s

- ➊ Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
- ➋ Emulate a random execution of $A'^{\text{Sign}'_{s'}}$ with a single twist:
 - On the i^* 'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather than choosing it via Gen)
 - When need to sign using s_{i^*} , use Sign_s .
- ➌ Let $(m, \sigma = (m_1, v_1, \sigma_1), \dots, (m_q, v_q, \sigma_q)) \leftarrow A'$
- ➍ Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$)

Definition of A

Algorithm 28 (A)

Input: $v, 1^n$

Oracle: Sign_s

- 1 Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
 - 2 Emulate a random execution of $A'^{\text{Sign}'_{s'}}$ with a single twist:
 - On the i^* 'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather than choosing it via Gen)
 - When need to sign using s_{i^*} , use Sign_s .
 - 3 Let $(m, \sigma = (m_1, v_1, \sigma_1), \dots, (m_q, v_q, \sigma_q)) \leftarrow A'$
 - 4 Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$)
- Sign_s is called at most once

Definition of A

Algorithm 28 (A)

Input: $v, 1^n$

Oracle: Sign_s

- 1 Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
 - 2 Emulate a random execution of $A'^{\text{Sign}'_{s'}}$ with a single twist:
 - On the i^* 'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather than choosing it via Gen)
 - When need to sign using s_{i^*} , use Sign_s .
 - 3 Let $(m, \sigma = (m_1, v_1, \sigma_1), \dots, (m_q, v_q, \sigma_q)) \leftarrow A'$
 - 4 Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$)
- Sign_s is called at most once
 - The emulated game $A'^{\text{Sign}'_{s'}}$ has the “right” distribution.

Definition of A

Algorithm 28 (A)

Input: $v, 1^n$

Oracle: Sign_s

- 1 Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
- 2 Emulate a random execution of $A'^{\text{Sign}'_{s'}}$ with a single twist:
 - On the i^* 'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather than choosing it via Gen)
 - When need to sign using s_{i^*} , use Sign_s .
- 3 Let $(m, \sigma = (m_1, v_1, \sigma_1), \dots, (m_q, v_q, \sigma_q)) \leftarrow A'$
- 4 Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$)

- Sign_s is called at most once
- The emulated game $A'^{\text{Sign}'_{s'}}$ has the “right” distribution.
- A breaks $(\text{Gen}, \text{Sign}, \text{Vrfy})$ whenever $i^* = \tilde{i} > 1$.

Analysis of A

For any $n \in \mathcal{I}$

$$\begin{aligned} & \Pr[A(1^n) \text{ breaks } (\text{Gen}, \text{Sign}, \text{Vrfy})] \\ & \geq \Pr_{i^* \leftarrow [p=p(n)]}[i = \tilde{i}] \\ & \geq \frac{1}{p} \cdot \Pr[A' \text{ breaks } (\text{Gen}', \text{Sign}', \text{Vrfy}')] \geq \frac{1}{p(n)^2} \end{aligned}$$

“Somewhat”-Stateful Schemes

A one-time scheme $(\text{Gen}, \text{Sign}, \text{Vrfy})$, and $\ell = \ell(n) \in \omega(\log n)$

Construction 29

- $\text{Gen}'(1^n)$: output $(s_\lambda, v_\lambda) \leftarrow \text{Gen}(1^n)$.
- $\text{Sign}'_s(m)$: choose *unused* $\bar{r} \in \{0, 1\}^\ell$
 - ① For $i = 0$ to $\ell - 1$: if $a_{\bar{r}_1, \dots, i}$ was not set:
 - ① For both $j \in \{0, 1\}$, let $(s_{\bar{r}_1, \dots, i, j}, v_{\bar{r}_1, \dots, i, j}) \leftarrow \text{Gen}(1^n)$
 - ② $\sigma_{\bar{r}_1, \dots, i} = \text{Sign}_{s_{\bar{r}_1, \dots, i}}(a_{1, \dots, i} = (v_{\bar{r}_1, \dots, i, 0}, v_{\bar{r}_1, \dots, i, 1}))$
 - ② Output $(\bar{r}, a_\lambda, \sigma_\lambda, \dots, a_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}} = \text{Sign}_{s_{\bar{r}}}(m))$
- $\text{Vrfy}'_v(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \dots, a_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}}))$
 - ① Verify $\text{Vrfy}_{v_{\bar{r}_1, \dots, i}}(a_{\bar{r}_1, \dots, i}, \sigma_{\bar{r}_1, \dots, i}) = 1$ for every $i \in \{0, \dots, \ell - 1\}$
 - ② Verify $\text{Vrfy}_{v_{\bar{r}}}(m, \sigma_{\bar{r}}) = 1$ (where $v_{\bar{r}} = (a_{\bar{r}})_{\bar{r}[\ell]}$)

1 More efficient scheme

- 1 More efficient scheme
- 2 Sign' does not keep track of the message history.

Somewhat-Stateful Schemes

- 1 More efficient scheme
- 2 Sign' does not keep track of the message history.
- 3 Each leaf is visited at most once.

Somewhat-Stateful Schemes

- 1 More efficient scheme
- 2 Sign' does not keep track of the message history.
- 3 Each leaf is visited at most once.
- 4 Each one-time signature is used once.

Lemma 30

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Proof:

Lemma 30

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Proof: Let $(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \dots, a_{\bar{r}-1}, \sigma_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}}))$ be the output of a cheating A' and let $a_{\bar{r}} = m$

Lemma 30

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is one time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ is a stateful existential unforgeable signature scheme.

Proof: Let $(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \dots, a_{\bar{r}-1}, \sigma_{\bar{r}_1, \dots, \ell-1}, \sigma_{\bar{r}}))$ be the output of a cheating A' and let $a_{\bar{r}} = m$

Claim 31

Whenever A' succeeds, $\exists \tilde{i} = \tilde{i}(m, \sigma') \in \{0, \dots, \ell\}$ such that:

- ① Sign'_s queried $\text{Sign}_{s_{\bar{r}_1, \dots, i}}(a_{\bar{r}_1, \dots, i})$ for every $i \in [\tilde{i} - 1]$, where $s_{\bar{r}_1, \dots, i}$ is the value sampled by Sign' when sampling $a_{\bar{r}_1, \dots, i-1}$ (or s_λ , if $\tilde{i} = 0$)
- ② Sign'_s did not query $\text{Sign}_{s_{\bar{r}_1, \dots, i}}(a_{\bar{r}_1, \dots, i})$.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

- 1 $\text{Gen}'(1^n) : \text{let } (s, v) \leftarrow \text{Gen}(1^n) \text{ and } \pi \leftarrow \Pi_{\ell(n), q(n)}, \text{ where } q \in \text{poly} \text{ is large enough for the application below, and outputs } (s' = (s, \pi), v' = v)$

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

- 1 $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n), q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
- 2 $\text{Sign}'(1^n)$:
 - 1 choose $\bar{r} = \pi(0^\ell \circ m)_{1, \dots, \ell}$

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

- 1 $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
- 2 $\text{Sign}'(1^n)$:
 - 1 choose $\bar{r} = \pi(0^\ell \circ m)_{1,\dots,\ell}$
 - 2 When setting $(s_{\bar{r}_{1,\dots,i,j}}, v_{\bar{r}_{1,\dots,i,j}}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\dots,i,j})$ as the randomness for Gen.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

- 1 $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
- 2 $\text{Sign}'(1^n)$:
 - 1 choose $\bar{r} = \pi(0^\ell \circ m)_{1,\dots,\ell}$
 - 2 When setting $(s_{\bar{r}_{1,\dots,i,j}}, v_{\bar{r}_{1,\dots,i,j}}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\dots,i,j})$ as the randomness for Gen.
- Sign' keeps no state

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

- 1 $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
 - 2 $\text{Sign}'(1^n)$:
 - 1 choose $\bar{r} = \pi(0^\ell \circ m)_{1,\dots,\ell}$
 - 2 When setting $(s_{\bar{r}_{1,\dots,i,j}}, v_{\bar{r}_{1,\dots,i,j}}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\dots,i,j})$ as the randomness for Gen.
- Sign' keeps no state
 - A single one-time signature key might be used several times, but always on *the same* message

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

- 1 $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
 - 2 $\text{Sign}'(1^n)$:
 - 1 choose $\bar{r} = \pi(0^\ell \circ m)_{1,\dots,\ell}$
 - 2 When setting $(s_{\bar{r}_{1,\dots,i,j}}, v_{\bar{r}_{1,\dots,i,j}}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\dots,i,j})$ as the randomness for Gen.
- Sign' keeps no state
 - A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme:

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0, 1\}^*$ to $\{0, 1\}^q$.

- ① $\text{Gen}'(1^n)$: let $(s, v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \text{poly}$ is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
- ② $\text{Sign}'(1^n)$:
 - ① choose $\bar{r} = \pi(0^\ell \circ m)_{1,\dots,\ell}$
 - ② When setting $(s_{\bar{r}_{1,\dots,i,j}}, v_{\bar{r}_{1,\dots,i,j}}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\dots,i,j})$ as the randomness for Gen.
- Sign' keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme: use PRF

Without CRH

Definition 32 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's A_1, A_2 :

$$\Pr[(x, a) \leftarrow A_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow A_2(a, h): \\ x \neq x' \wedge h(x) = h(x')] = \text{neg}(n)$$

Without CRH

Definition 32 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's A_1, A_2 :

$$\Pr[(x, a) \leftarrow A_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow A_2(a, h): \\ x \neq x' \wedge h(x) = h(x')] = \text{neg}(n)$$

Theorem 33

OWFs imply efficient compressing TCRs.

Definition 34 (target one-time signatures)

A signature scheme $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is target one-time existential unforgeable (for short, target one-time signature), if for any pair of PPT's A_1, A_2

$$\Pr[(m, a) \leftarrow A_1(1^n); (s, v) \leftarrow \text{Gen}(1^n); \\ (m', \sigma) \leftarrow A_2(a, \text{Sign}_s(m)): m' \neq m \wedge \text{Vrfy}_v(m', \sigma) = 1] \\ = \text{neg}(n)$$

Definition 34 (target one-time signatures)

A signature scheme $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is target one-time existential unforgeable (for short, target one-time signature), if for any pair of PPT's A_1, A_2

$$\Pr[(m, a) \leftarrow A_1(1^n); (s, v) \leftarrow \text{Gen}(1^n); \\ (m', \sigma) \leftarrow A_2(a, \text{Sign}_s(m)): m' \neq m \wedge \text{Vrfy}_v(m', \sigma) = 1] \\ = \text{neg}(n)$$

Claim 35

OWFs imply target one-time signatures.

Definition 36 (random one-time signatures)

A signature scheme $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is random one-time existential unforgeable (for short, random one-time signature), if for any PPT A and any samplable ensemble $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$, it holds that

$$\begin{aligned} & \Pr[m \leftarrow \mathcal{M}_n; (s, v) \leftarrow \text{Gen}(1^n); (m', \sigma) \leftarrow A(m, \text{Sign}_s(m)) : \\ & \quad m' \neq m \wedge \text{Vrfy}_v(m', \sigma) = 1] \\ & = \text{neg}(n) \end{aligned}$$

Definition 36 (random one-time signatures)

A signature scheme $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is random one-time existential unforgeable (for short, random one-time signature), if for any PPT A and any samplable ensemble $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$, it holds that

$$\begin{aligned} & \Pr[m \leftarrow \mathcal{M}_n; (s, v) \leftarrow \text{Gen}(1^n); (m', \sigma) \leftarrow A(m, \text{Sign}_s(m)) : \\ & \quad m' \neq m \wedge \text{Vrfy}_v(m', \sigma) = 1] \\ & = \text{neg}(n) \end{aligned}$$

Claim 37

Assume $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is target one-time existential unforgeable, then it is random one-time existential unforgeable.

Lemma 38

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is a target one-time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ from Construction 29 is a stateful existential unforgeable signature scheme.

Lemma 38

Assume that $(\text{Gen}, \text{Sign}, \text{Vrfy})$ is a target one-time signature scheme, then $(\text{Gen}', \text{Sign}', \text{Vrfy}')$ from Construction 29 is a stateful existential unforgeable signature scheme.

Proof: ?